AIR Project Summary Report

José Rufino (DI-FCUL)
Sérgio Filipe (Skysoft Portugal, SA)

DI-FCUL TR–07–36

December 2007

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749-016 Lisboa
Portugal

Abstract

This document summarizes the main results of AIR, an innovation initiative sponsored by ESA, the European Space Agency. The acronym AIR stands for ARINC 653 Interface in RTEMS. The ARINC 653 is a civil aviation world specification addressing safety critical and certification issues in embedded systems software. The AIR Project studied the adoption of ARINC 653 in space on-board software together with the utilization of RTEMS, the Real-Time Executive for Multiprocessor Systems.

This document addresses: (i) the AIR architecture specification; (ii) the AIR support to partitioning mechanisms; (iii) the mapping of ARINC 653 services into RTEMS; (iv) the proof of concept prototypes.

AIR

ARINC 653 Interface in RTEMS

Summary Report

	Name	Title	Signature	Date
Written	Sérgio Filipe	Technical Manager		20-09-2007
Written	José Rufino	Senior Researcher		20-09-2007
Verified	José Rufino	Senior Researcher		20-09-2007
Verified	José Neves	Project Manager		20-09-2007
Approved				

Issue	Date	Description	Author
1.0	20-09-2007	ARINC 653 Interface in RTEMS – Summary Report	Sérgio Filipe, José Rufino

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to provide a summary report of the work performed on the AIR (ARINC 653 Interface in RTEMS) project.

1.2 Scope

This document defines the AIR Project Summary Report deliverable.

1.3 Applicable Documents

Reference	Title
[AIRPROP]	AIR Proposal, Skysoft, September 2005
[TMREPLY]	Technical Memo for the AIR Proposal (negotiation points replies)
[AIRMMIN]	Minutes of Meeting (From Project's Kick-Off Meeting to the Sixth Internal Meeting)
[AIRWP1]	AIR WP1 Output: Requirements, Architecture and Services
[AIRWP2]	AIR WP2 Output: Overall System Specification
[AIRWP3]	AIR WP3 Output: AIR Design Results and Proof of Concept

Table 1 – Applicable Documents

1.4 Document Outline

Section 1 Introduces the document by indicating its purpose and scope, its content and which documents it refers to.

Section 2 Describes the AIR architecture.

Section 3 Describes the AIR proof of concept demonstrators implemented

Section 4 Concludes the document.

Annex A Summarizes a set of AIR Project Result indicators.

1.5 Document Contributions

This document was jointly prepared by Faculdade de Ciências da Universidade de Lisboa (FCUL) and Skysoft Portugal.

1.6 Acronyms and Abbreviations

Acronym	Definition
AEEC	Airlines Electronic Engineering Committee
AIR	ARINC 653 Interface in RTEMS
APEX	Application Executive
ARINC	Aeronautical Radio, Inc.
API	Application Programming Interface
BSP	Board Support Package
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
DDD	Data Display Debugger
eCos	Embedded Configurable Operating System
EUROCAE	European Organization for Civil Aviation Equipment
FAA	Federal Aviation Administration
FCUL	Faculdade de Ciências da Universidade de Lisboa
FIFO	First In First Out
GCD	Greatest Common Divisor
GDB	GNU Debugger
GNU	GNU's Not Unix
GRUB	GRand Unified Bootloader
HAL	Hardware Abstraction Layer
IMA	Integrated Modular Avionics
ISR	Interrupt Service Routine
MEC	Multi-Executive Core
MILS	Multiple Independent Levels of Security
MMU	Memory Management Unit
MTF	Major Time Frame
OS	Operating System
PMK	Partition Management Kernel
PST	Partition Scheduling Table
PTD	Page Table Descriptor
PTE	Page Table Entry
RAM	Random Access Memory
RTCA	Radio Technical Commission for Aeronautics
RTEMS	Real-Time Executive for Multiprocessor Systems
RTOS	Real-time Operating System
SEC	Single-Executive Core
SIS	SPARC Instruction Simulator

Table 2 – Acronyms and Abbreviations

2 AIR Architecture Definition

This section present the AIR architecture defined on the context of the AIR activity

2.1 Defining the AIR Architecture

The fundamental architecture definition of the AIR system, advanced in [AIRWP1], thoroughly discussed in [AIRWP2] and fully defined on [AIRWP3], is sketched in the diagram of Figure 1. It makes use of:

- a multi-executive core software layer;
- a two-level hierarchical scheduler.

The multi-executive core approach allows the use of a different RTOS kernel instance per partition. That means different RTOS kernels may be used at each partition. Even if the RTOS integration is homogeneous, i.e. even if the same RTOS kernel (e.g. RTEMS) is being used in every partition, these should be regarded as different RTOS kernels, with its own process scheduler. This structure allows the implementation of the hierarchical scheduler design approach.

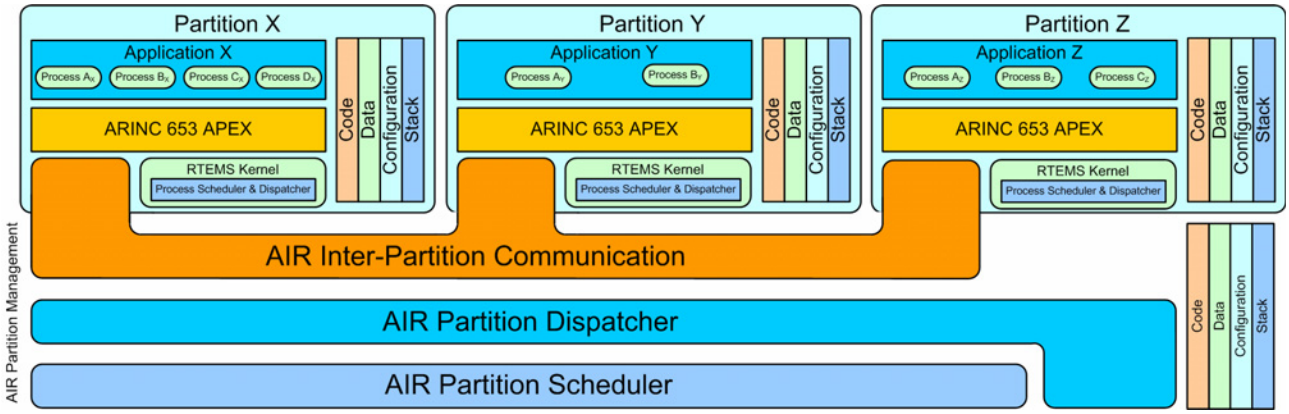

Figure 1 – Overview of the AIR System Architecture

The integration of the different components in the architecture of Figure 1 assumes a well-defined specification of their interfaces in order to preserve the modularity characteristics of the architecture. The components added to the native RTOS kernel aim to supply the functionality missing in those operating systems, to be in conformity with the standard ARINC 653 specification. The fundamental AIR design components integrated in the architecture of Figure 1 are the following:

- **AIR partition scheduler**, selecting at given times which partition owns system resources, namely the processing infrastructure. It secures temporal segregation using a single fixed cyclic scheduler.
- **AIR partition dispatcher**, which has the responsibility of saving the execution context of the running partition and of restoring the execution context for the heir partition. It secures the management of all provisions required to guarantee spatial segregation.
- **AIR inter-partition communication module**, allowing the exchange of information between different partitions without violating spatial segregation constraints.
- he **native RTOS kernel**, which in conformity with the architectural attributes of the multi-executive core design can be specifically configured for each partition [AIRWP2];
- the system partitions doing an additional use of **system specific OS functions**, in general supplied also with the native RTOS distribution and that, again, may be individually configured in a per-partition basis;
- **ARINC 653 application executive interface (APEX)**, for each partition in the system, defining the services in strict conformity with the ARINC 653 standard.

The **application executive (APEX) interface** should be designed as much as possible by mapping the ARINC 653 service primitives into the service interface of the native RTOS kernel, i.e. into the native RTEMS primitives [RTEMSAPI], for the particular case of the current AIR architecture. As a design alternative, other OS interfaces may be used, such as the POSIX interface, also offered by RTEMS [RTEMSPOSIX].

The use of highly configurable **RTOS kernels** allows an effective adaptation of the services provided by the RTOS kernel to the requirements of each partition. This is especially true for system partitions, where in addition to the standard application executive (APEX) interface there is a need for system specific OS functions.

The **system specific OS functions** in essence establish a set of interface stubs to address the specific details of each hardware platform. In conformity with the ARINC 653 standard specification, this functionality is only allowed in system partitions.

2.2 The AIR Partitioning Support Mechanisms

2.2.1 Design Principles

To ensure flexibility and modularity, instead of modifying the RTOS scheduler to extend it to the partitioning concept, the approach followed in the AIR architecture uses one instance of the native RTOS scheduler (as provided by the RTEMS kernel, in the example illustrated in Figure 1) for process priority-based preemptive scheduling inside each partition. This is in conformity with the ARINC 653 specification. No fundamental modification is needed to the functionality of the RTOS process scheduler for its integration in the AIR system. In fact, this two-level hierarchical scheduler approach secures partition and process scheduler decoupling, thus allowing the use of different operating systems in different partitions (e.g. RTEMS, eCos,...).

2.2.2 Dependability Attributes

The use of a RTOS kernel instance per partition is also useful to guarantee a set of safety and security-related attributes, given it restricts code, data, configuration and execution context references to the confined and protected scope of a partition (cf. Figure 1).

2.2.3 Securing Spatial Segregation

An effective support to ARINC 653 spatial partitioning requires the use of specific memory protection mechanisms usually implemented in a hardware memory management unit (MMU). This comprises not only the protection of partition memory addressing spaces but also a functional protection concerning the management of privilege levels and restrictions to the execution of privileged instructions. Though there is room for enhancements, a basic set of such mechanisms do exist in the Intel IA-32 architecture and, in a given extent, in the SPARC LEON and ERC32 processor cores. Inside each partition, i.e. at the process level, a flat memory addressing scheme fully compliant with the ARINC 653 requirements is specified for the AIR architecture.

2.2.4 Securing Temporal Segregation

The ARINC 653 standard specification restricts the processing time assigned to each partition, in conformity with given configuration parameters. This means a single partition cannot monopolize the usage of the processor infrastructure thus preventing the applications in other partitions of being executed.

The scheduling of partitions defined by the ARINC 653 standard is strictly deterministic over time. Each partition has a fixed temporal window in which it has control over the computational platform. Each partition is scheduled on a fixed, cyclic basis. A Major Time Frame (MTF) of fixed duration, defined off-line, is periodically repeated throughout runtime operation. In the AIR architecture, temporal segregation is ensured by the AIR partition scheduler.

2.3 AIR APEX Interface Implementation

This section summarizes the implementation of the ARINC 653 APEX functionality onto RTEMS in the sequence of the AIR requirements and architecture expressed in [AIRWP1]; the AIR overall system specification expressed in [AIRWP2] and the proof of concept software demonstrator implemented in [AIRWP3].

The demonstrator is centered on the interaction of a number of processes within a single partition. Therefore this implementation composes, in this scenario, a relevant subset of the functionality specified in [AIRWP2], including the APEX process and partition management services and a number of intrapartition services.

The full design of the APEX implementation over RTEMS can be found on [AIRWP2] deliverable. In this summary we will only refer the implemented functionalities.

2.3.1 Development Environment

The APEX interface development environment consisted on a set of interacting Open Source software tools, deployed on an Intel x86-based desktop PC.

TOOLS	Description	Role
Eclipse IDE	Fedora Core Release	software development environment
rtems GCC	OAR corporation gcc-3.2.3-20040420	rtems backend GCC compiler
Fedora Core 6	Linux based OS	host OS
RTEMS	4.6.6 Release	target OS

Table 3 – APEX Interface development environment

2.3.1.1 AIR Partition Management

Partition management module does not exactly match the functional description of the ARINC 653 **partition concept**, in the sense it effectively manages partition information and data, but the *real weight* of enforcing time and space segregation is left to the PMK modules.

Implemented Services

`SET_PARTITION_MODE` is implemented and used in the proof of concept demonstrator initialization phase. It currently supports setting the partition mode to **`NORMAL`**.

2.3.1.2 AIR Process Management

Process management encompasses key APEX functionality concepts that are defined in terms of an already existing system, in this case the RTEMS core and its native API [RTEMSAPI].

AIR Process definition

The ARINC 653 process definition is implemented as a RTEMS task, as it is defined in RTEMS native API.

AIR Process scheduling

The ARINC 653 standard imposes a **preemptive priority-based scheduling** algorithm which ensures that the process which is executing on the processor at any point in time is the one with the highest priority among all processes in the ready state with FIFO order applied to processes with the same priority.

RTEMS own scheduling concepts do match the APEX scheduling rules. The followed approach uses one instance of the RTEMS scheduler for process scheduling inside each partition. **No fundamental modification** is needed to the functionality of this component for its integration in the AIR system architecture.

AIR Process state

The states of ARINC 653 processes and RTEMS tasks do not exactly match. There exist in RTEMS no transition to the **DORMANT** state (RTEMS has no **STOP** directives). These transitions are thus emulated into matching the functionality specified for the APEX by AIR proof of concept implementation.

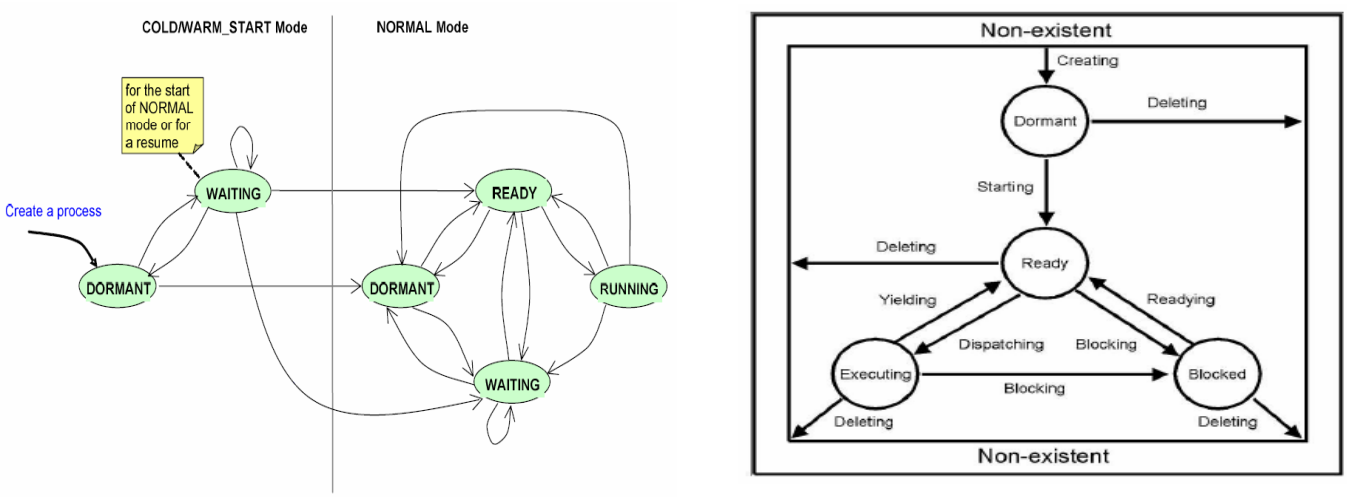

Figure 2 – ARINC 653 process states (on the left) vs RTEMS task states

(NOTE: figures adapted from [A563P2] and [RTEMSAPI])

Implemented Services

```
CREATE_PROCESS
START
STOP
SUSPEND
RESUME
GET_MY_ID
GET_PROCESS_ID
GET_PROCESS_STATUS
```

Here above is the relevant sub-set of the implemented services both those that are relevant to cover the process control needs as also those that are used in the proof of concept demonstrator.

2.3.2 AIR Blackboard services

Blackboard services as specified on ARINC 653 standard provide functionalities for intra-partition communication. In contrast to buffer, blackboard does not support a message queue: when a message is displayed it overwrites the previous one. This unique shared message can be cleared at any time by any process on the same partition (in a similar nature to the functionality of sampling ports).

If a message is available at the blackboard, then any process can access it instantly with no reserve. Processes only block on a blackboard when trying to read it while the blackboard is empty. When it becomes not empty, processes are automatically unblocked.

Since there is no similar RTEMS object to which the APEX Blackboard can be mapped to directly, this object shall be defined using a global variable (that shall hold the blackboard message) with access controlled by mutexes and conditions.

Implemented services

```
CREATE_BLACKBOARD
DISPLAY_BLACKBOARD
READ_BLACKBOARD
CLEAR_BLACKBOARD
GET_BLACKBOARD_ID
GET_BLACKBOARD_STATUS
```

All the Blackboard service requests are used in the proof of concept demonstrator, allowing for the communication of processes within a partition. These services are also representative of the overall intra partition services.

2.4 Summary of AIR Architecture Attributes

The AIR architecture is built from a **multi-executive core two-level hierarchical multi-scheduler solution**. The engineering of this solution requires an effective integration of AIR components with the RTOS source tree and with the corresponding application production chain.

The architectural attributes of the multi-executive core design in respect to the definition of a service interface conformant with the ARINC 653 standard, using a commercial off-the-shelf RTOS kernel such as RTEMS, is summarized in Table 4.

Multi-Executive Core Architecture Attribute								
RTOS Integration	**Versatility**	**Modularity**	**Configurability**	**Integrity**	**Fault Confinement**	**Footprint Size**	**Bootstrap Method**	**Development Tools**
partitioned	good	good	good	good	good	non optimal	single-file	canonical plus object filtering
							multi-file	canonical plus bootstrap tool

Table 4 – Summary of AIR Multi-Executive Core Architecture Attributes

3 AIR Proof of Concept Prototypes

This section briefly describes the proof of concept prototypes developed for and presented in the final review of the AIR Project.

3.1 Prototyping Environment

The proof of concept prototypes were built using the RTEMS 4.6.6 version enhanced with a graphical window manager, dubbed VITRAL (VITRAL is the Portuguese word for stained glass window) developed within other projects [COUT06a] but also considered useful in the context of the AIR Project. The utilization of an application using VITRAL is illustrated in Figure 3.

Figure 3 – The VITRAL visualization environment as used in the AIR proof of concept prototypes

The VITRAL window manager is built for real-time embedded systems. As such, it has little memory footprints and time-bounded functionality. The hardware utilization requirements are minimal: in the essence it merely requires a support for a standard VGA video interface.

The AIR Multi-Executive Core (MCE) proof of concept demonstrator illustrates partition and task scheduling using multiple RTEMS kernels (one per partition and another for the support of the AIR PMK Partition Manager). A possible assignment of functional components to partitions in a space application is exemplified in the table of Figure 4. The scheduling of those partitions over a MTF (Major Time Frame) is also illustrated in Figure 4. Each partition is composed by a set of periodic tasks which print a string to a VITRAL window, as shown Figure 3

Partition	Function
Partition X	Attitude Control
Partition Y	Telemetry Tracking and Command
Partition Z	On-Board Data Handling
System Partition	Communications

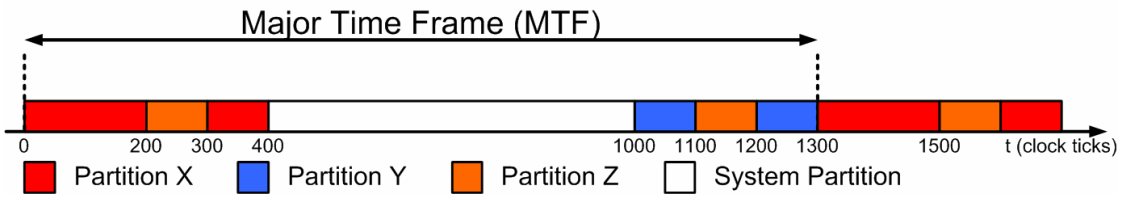

Figure 4 - Example of Partition function assignment and scheduling over a Major Time Frame

3.2 APEX Interface Demonstrator

The APEX Interface Demonstrator does not aim to demonstrate the total compliance of AIR implementation with the ARINC 653 standard or to exhaustively test its correct functionality. Instead, the purpose of this demonstrator is to show evidence that basic APEX features can be implemented over RTEMS and that **correct** behaviour is achieved. As such it provides a valid evidence to advance for a full APEX implementation over RTEMS.

As to implement the APEX Interface Demonstrator the following APEX resources were used:

- **Process P1** – this is the main process that sends messages to the blackboards. It has the lowest priority of the three processes used;
- **Process P2** – this process waits for a message to be available on blackboard BB2. When this condition is met, it reads the message and clears the blackboard. It then reacts according with the received message. Its priority his above P1.
- **Process P3** – this process is similar to P2 but uses blackboard BB3 instead of blackboard BB2. Also, it has the same priority as P2.
- **Blackboard BB2** – this blackboard shall be used for P1 to send messages to P2;
- **Blackboard BB3** – this blackboard shall be used for P1 to send messages to P3.

The demonstrator main focus relies on the following issues using the listed above resources:

- Basic operations over APEX processes like **`CREATE`**, **`START`**, **`STOP`**, **`SUSPEND`**, **`RESUME`** using RTEMS tasks to implement APEX processes;
- Intra partition communication between AIR processes by using the APEX Blackboard service;
- The implementation of the APEX priority driven process scheduler by using the RTEMS native task scheduler

4 Conclusions and Future Challenges

This document provides a summary report of the work developed and achievements of the AIR activity.

The AIR architecture aims to provide the developers and the integrators of space on-board software with an environment that is standard and in strict conformity with the ARINC 653 specification [1]. The AIR solution is hardware and operating system independent and it exploits the usage of conventional off-the-shelf license-free open-source RTOS kernels, such as RTEMS, a real-time multitasking kernel qualified for use in space on-board software developments.

A specific AIR Partition Management Kernel (PMK) incorporates the fundamental functionality needed for conformity with the ARINC 653 standard specification. A logical connection to a set of RTOS kernel instances provides the remaining functional requirements.

The AIR architecture specification also addresses the fundamental issues concerning the enforcement of spatial segregation, through memory protection mechanisms, and temporal segregation, through a partition scheduling fixed cyclic policy.

The mapping task was a far more complex than expected due to the some great and unexpected differences between RTEMS and the ARINC 653 standards. The main differences founds were:

- tasks states of RTEMS and process states of ARINC 653 do not match;
- ARINC 653 process are always in memory while RTEMS tasks are deleted from memory when stopped;
- the lack of functionality in RTEMS to provide at any given moment the task status (it only has a task stating if it is suspended or not);
- the implementation of the periodic processes using the rate monotonic manager was not possible with the current functionalities of the referred manager

These differences implied extra analysis time and added complexity on the built specification.

The APEX Implementation Demonstrator enables the visualization of APEX processes being scheduled and communicating with each other.

Given this, the APEX proof of concept Demonstrator provides good evidence that it is feasible to fully implement the APEX interface over RTEMS. However as to achieve a fully functional and 100% compliant implementation a hard work is still foreseen and some issues might represent hard issues to solve.

ANNEX A AIR RESULTS SUMMARY

This annex summarizes the main results produced within the scope of the **AIR Project – ARINC 653 Interface in RTEMS**.

A.1 AIR DELIVERABLE REPORTS

Work Package	Title	Pages
[AIRWP1]	AIR Requirements, Architecture and Services	63
[AIRWP2]	AIR Overall System Specification	144
[AIRWP3]	AIR Design Results and Proof of Concept	70
[AIRWP3]	AIR Final Report	57
[AIRWP3]	AIR Summary Report	14

Table 5 – AIR Deliverable Reports

A.2 AIR PROTOTYPE DEMONSTRATORS

Work Package	Proof of Concept Demonstrator	Project Usefulness
[AIRWP2-P]	AIR PMK Single Executive Core (SEC)	Validation of fundamental AIR design issues
[AIRWP3-P]	AIR PMK Multi-Executive Core (MEC)	Validates the AIR overall system architecture design
[AIRWP3-P]	AIR APEX Interface	A proof of concept implementation of the APEX over the RTEMS API.

Table 6 – AIR Prototype Demonstrators

A.3 AIR CURRENT PUBLICATIONS

Papers in Conferences or Workshops		
Event	**Title**	**Authors**
DASIA 2007[1]	ARINC 653 Interface in RTEMS	José Rufino, Sérgio Filipe, Manuel Coutinho, Sérgio Santos, James Windsor.

Table 7 – AIR Current Publications

[1] DASIA 2007 – Data Systems in Aerospace, Naples, Italy, May-June 2007